Project Details

Name: Speedway RFID Swimming Prototype
Researcher: Richard McCarthy
Date: 8th May 2020

Work Carried Out

A detailed view of various aspects of a technical architecture was carried out.

Basic overview diagrams are provided below as the full details to write out and construct in software diagrams would take another 1 to 2 days. This time is better spent starting on the development of the software now and set time aside for the roadmap to put those additional details in place at that point.

A significant amount of time has been spend on assessing the best design for portions of the cloud application bearing heavily in mind the required ability to scale and extend the functionality of the application in the future.

Given this and considering the nature of data that will be attempted to capture from the RFID reader, a real time distributed data streaming system has been decided as the best approach. This allows it to be a system that can scale and transform data all in real time into more meaningful metrics coupled with the ability to be extended for potential use in analytics intelligence in the future.

Apache Kafka is a distributed streaming platform that can sit at the heart of such a system and provide the ability to deal with potentially vast volumes of data with low latency processing times and allows easy deployment on any system.

It has the ability to transform data in real time which will allow the creation of different “views” which can be sent in real time to a client application, store those “views” and also the original streaming data is stored so there is an audit log that can be replayed. This audit log opens the door for other analytical uses later on.

The Java standalone application that sits in the swimming pool will contain the algorithm to filter and association “*action states*” with the data to infer laps.
Each piece of filtered data can be sent to the cloud where the streaming data can be transformed and potentially other domain specific intelligence can then be applied to it to extra further the data for display.

The client application for displaying the swimming data would be “subscribed” to the cloud and each piece of transformed data from the streaming service would be pushed in real time to the client application screen.

A future version of the client application could contain log in with set up details for various swimmers being attached to tags.A section for a coach to load a session so a session identified and a location identified could be attached to the data that is coming from the reader.

In such a scenario the Java application in the swimming pool would first contact the cloud to authenticate itself, then request details of the next swimming session if they exist and any other relevant details that were attached.
At this point it would be ready to start reading data and sending the filtered data to the cloud for further processing.

The details of how to store the data has not been fully decided but this will not hold up the start of development and the best approach for this system can be decided during development.
The 2 options here are -

1) to use the storage that comes in-built with the Apache Kafka streaming platform, which is essentially a database underneath as it is a stored commit log of everything. It is far more than just a messaging system like RabbitMQ or ActiveMQ. It is used in this manner by many of the top streaming data processing sites for media, travel and banking.
2) that as each piece of data is transformed in real time from the streaming data in Apache Kafka and then stored n a standard (more familiar) external database.

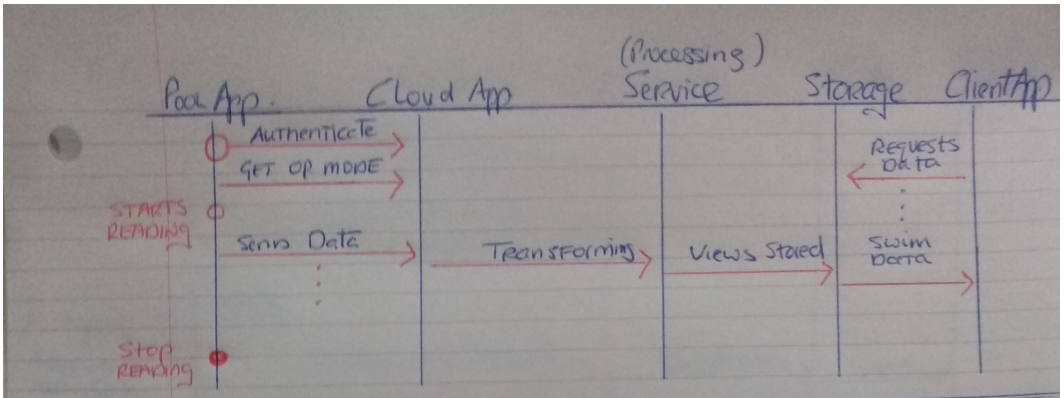
All other aspects of the system will be using the same technologies as provided in an earlier report, namely -

Java Application - Pure Java 8
Cloud Application - Spring Boot, Java, Apache Kafka, AWS, (maybe MongoDB also)
Client Application - AngularJS/Typescript/HTML (Single Page Application)

All designed using a *Microservices* architecture centred around *EventSourcing* and *CQRS* design patterns.

The system will contain quite a lot of complexity to deal with real time data processing and storage so this will take up the remaining time to get the outline built and capture and process the essential data.

The benefit is once the system is in place it allows for many more “plug-in” options and opens up to additional complexity without complete re-works. The suite of technologies used are very much leading industry approached that are well established and used by some of the most data intensive companies around so their support and ongoing development and skill-set available are very much assured.

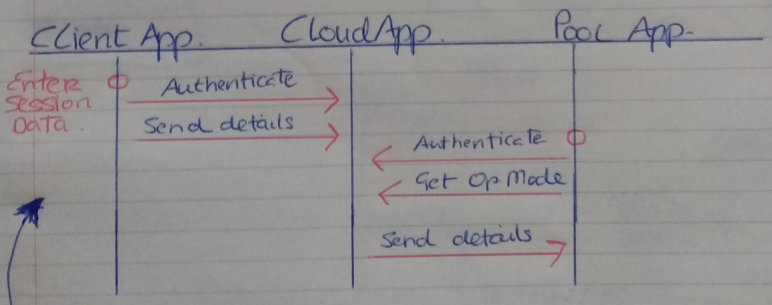


Example of Data Record.

- EPC
- Time
- RSSI.
- ReadRate
- Selected.
- ActionState
- SessionId
- Location

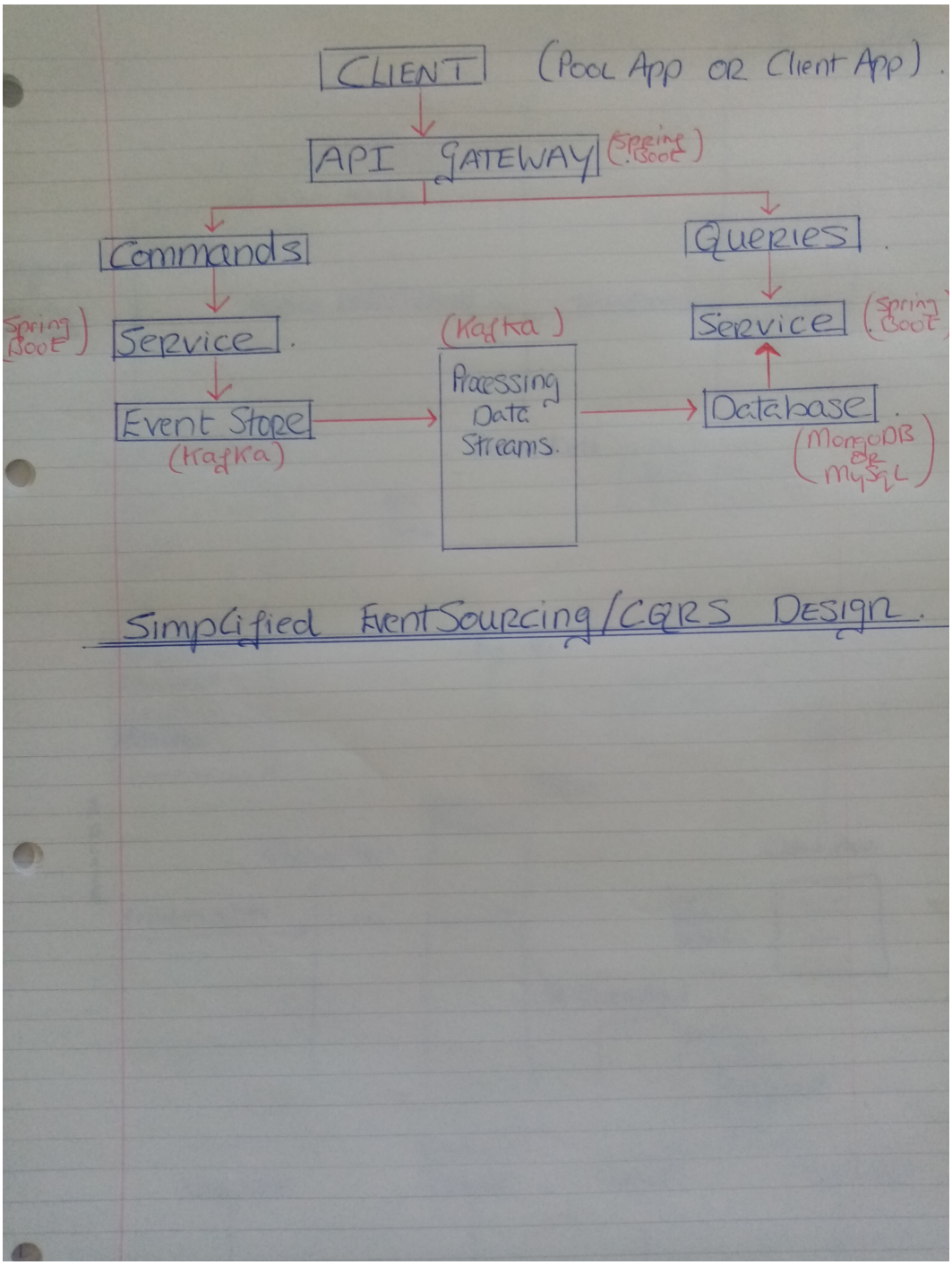
← A stream of these records would be processed & transformed into more client meaningful data.

A full audit trail of the filtered data could be stored also for later use.



Future version where client could enter in swim session details, which are saved to the cloud for a specific location reader.

The pool app sends 'Get Op Mode' which could be used to help decide the structure of a swim session and might adjust the filtering algorithm automatically.
e.g. No dive, multiple divers etc. . .



Simplified EventSourcing/CQRS Design.

Known Blockers

Currently none

Next Steps

Starting building the standalone Java application that will take data from the reader and aim to filter it into meaningful captured swimming data.